Sparse Gaussian Processes Using Backward Elimination

Liefeng Bo, Ling Wang, and Licheng Jiao

Institute of Intelligent Information Processing and
National Key Laboratory for Radar Signal Processing,
Xidian University, Xi'an 710071, China
{blf0218, wliiip}@163.com

Abstract. Gaussian Processes (GPs) have state of the art performance in regression. In GPs, all the basis functions are required for prediction; hence its test speed is slower than other learning algorithms such as support vector machines (SVMs), relevance vector machine (RVM), adaptive sparseness (AS), etc. To overcome this limitation, we present a backward elimination algorithm, called GPs-BE that recursively selects the basis functions for GPs until some stop criterion is satisfied. By integrating rank-1 update, GPs-BE can be implemented at a reasonable cost. Extensive empirical comparisons confirm the feasibility and validity of the proposed algorithm.

1 Introduction

Covariance functions have a great effect on the performance of GPs. The experiments performed by Williams [1] and Rusmussen [2] have shown that the following covariance function works well in practice

$$C(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\sum_{p=1}^{d} \theta_p \left(x_i^p - x_j^p \right)^2 \right) \tag{1.1}$$

where θ_p is scaling factor. If some variable is unimportant or irrelevant for regression, the associated scaling factor will be made small; otherwise it will be made large.

The key advantage of GPs is that the hyperparameters of covariance function can be optimized by maximizing the evidence. This is not appeared in other kernel based learning methods such as support vector machines (SVMs) [3]. In SVMs, an extra model selection criterion, e.g. cross validation score is required for choosing hyperparameters, which is intractable when a large number of hyperparameters are involved. Though GPs are very successful, they also have some shortages: (1) the computational cost of GPs is $O(l^3)$, where l is the size of training samples, which seems to prohibit the applications of GPs to large datasets; (2) all the basis functions are required for prediction; hence its test speed is slower than other learning algorithms such as SVMs, relevance vector machine (RVM) [4], adaptive sparseness (AS) [5], etc.

Some researchers have tried to deal with the shortages of GPs. In 2000, Smola et al. [6] presented sparse greedy Gaussian processes (SGGPs) whose computational cost is $O\left(kn^2 l\right)$, where n is the number of basis functions and k is a constant factor. In 2002, Csató et al. also proposed sparse on-line Gaussian processes (SOGPs) [7] that result in good sparseness and low complexity simultaneously. However, both SGGPs and SOGPs throw away the key advantage of GPs. As a result, they have difficulties in tackling the hyperparameters.

This paper focuses on the second shortage of GPs above. We propose a backward elimination algorithm (GPs-BE) that recursively selects the basis functions with the smallest leave-one-out score at the current step until some stop criterion is satisfied. GPs-BE has reasonable computational complexity by integrating rank-1 update formula. GPs-BE is performed after GPs is trained; hence all the advantages of GPs are reserved. Extensive empirical comparisons show that our method greatly reduces the number of basis functions of GPs almost without sacrificing the performance.

2 Gaussian Processes

Let $\mathbf{Z}=\left\{\left(\mathbf{x}_i, y_i\right)\right\}_{i=1}^{l}$ be l empirical samples set drawn from

$$y_i = f\left(\mathbf{x}_i, \mathbf{w}\right) + \varepsilon_i, \quad i = 1, 2, \cdots l \tag{2.1}$$

where ε_i is independent sample from some noise process which is further assumed to be mean-zeros Gaussian with variance σ^2. We further assume

$$f\left(\mathbf{x}, \mathbf{w}\right) = \sum_{i=1}^{l} w_i C\left(\mathbf{x}, \mathbf{x}_i\right) \tag{2.2}$$

According Bayesian inference, the posterior probability of $\mathbf{w}$ can be expressed as

$$P\left(\mathbf{w} \mid \mathbf{Z}\right) = \frac{P\left(\mathbf{Z} \mid \mathbf{w}\right) P\left(\mathbf{w}\right)}{P\left(\mathbf{Z}\right)} \tag{2.3}$$

Maximizing the log-posterior is equivalent to minimizing the following objective function

$$\hat{\mathbf{w}} = \arg\min\left(L\left(\mathbf{w}, \lambda\right) = \left(\mathbf{w}^T \left(\mathbf{C}^T \mathbf{C} + \sigma^2 \mathbf{I}\right) \mathbf{w} - 2\mathbf{w}\mathbf{C}^T \mathbf{y}\right)\right) \tag{2.4}$$

where $\mathbf{I}$ is the identity matrix. Hyperparameters are chosen by maximizing the following evidence

$$P\left(\boldsymbol{\theta}, \sigma^2 \mid \mathbf{Z}\right) = \left(2\pi\right)^{-l/2} \left|\sigma^2 \mathbf{I} + \mathbf{C}\mathbf{C}^T\right|^{-1/2} \exp\left(-\frac{1}{2} \mathbf{y}^T \left(\sigma^2 \mathbf{I} + \mathbf{C}\mathbf{C}^T\right)^{-1} \mathbf{y}\right) \tag{2.5}$$

In the related Bayesian models, this equality is known as the marginal likelihood, and its maximization is known as the type-II maximum likelihood method [8]. Williams [9] has demonstrated that this model is equivalent to Gaussian Processes (GPs) with the covariance $\left(\sigma^2 \mathbf{I} + \mathbf{C}\mathbf{C}^T\right)$; hence we call it GPs in this paper.

3 Backward Elimination for Gaussian Processes

In GPs, all the basis functions are used for prediction; therefore it is inferior to neural networks, SVMs and RVM in testing speed, which seems to prohibit its application in some fields. Here, GPs-BE is proposed to overcome this problem that selects the basis function by a backward elimination technique after training procedure. GPs-BE is a backward greedy algorithm that recursively removes the basis function with the smallest leave-one-out score at the current step until some stop criterion is satisfied.

For convenience of derivation, we reformulate (2.6) into

$$\mathbf{w} = \mathbf{H}^{-1}\mathbf{b} \tag{3.1}$$

where $\mathbf{H} = (\mathbf{C}^T\mathbf{C} + \sigma^2\mathbf{I})$ and $\mathbf{b} = \mathbf{C}^T\mathbf{y}$. Let $\Delta f^{(k)}$ be the increment of L with the k^{th} training sample deleted and then the following theorem holds true.

Theorem 3.1: $\Delta f^{(k)} = \frac{(w_k)^2}{\mathbf{R}_{kk}}$, where $\mathbf{R} = \mathbf{H}^{-1}$, $\mathbf{R}_{kk}$ denotes the k^{th} diagonal element of $\mathbf{H}^{-1}$.

We call $\Delta f^{(k)}$ leave-one-out score. At each step, we will remove the basis function with the smallest leave-one-out score. The index of the basis function to be deleted can be obtained by

$$s = \arg\min_{k\in P}\left(\Delta f^{(k)}\right), \tag{3.2}$$

where P is a set of the indices of the remainder basis functions. Note that the (l+1)-th variable, i.e. the bias, is preserved during the backward elimination process.

When one basis function is deleted, we require updating the matrix $\mathbf{R}$ and the vector $\mathbf{w}$. In terms of a rank-1 update, $\mathbf{R}^{(s)}$ and $w^{(s)}$ can be formulated as

$$\left(\mathbf{R}^{(s)}\right)_{ij} = \mathbf{R}_{ij} - \frac{\mathbf{R}_{is}\mathbf{R}_{sj}}{\mathbf{R}_{ss}}, \quad i, j \neq s, \tag{3.3}$$

$$\left(w^{(s)}\right)_i = \sum_{j\neq s}^{n}\left(\mathbf{R}_{ij} - \frac{\mathbf{R}_{is}\mathbf{R}_{sj}}{\mathbf{R}_{ss}}\right)b_j, \quad i \neq s\,. \tag{3.4}$$

Together with $\mathbf{w} = \mathbf{R}\mathbf{b}$, (3.4) is simplified as

$$\left(w^{(s)}\right)_i = w_i - w_s\frac{\mathbf{R}_{is}}{\mathbf{R}_{ss}}, \quad i \neq s\,. \tag{3.5}$$

Suppose that Δ_t is the increment of f at the t-th iteration, and then we will terminate the backward elimination procedure if

$$\Delta_t \le \varepsilon f \tag{3.6}$$

where we set $\varepsilon = 0.01$. The detailed backward elimination procedure is summarized in Figure 3.1.

```
Agorithm1: GPs-BE
1. Compute the index of basis function to be removed by (3.2);
2. Update the matrix R and the vector w by (3.3) and (3.5);
3. Remove the index resulting from step 1;
4. If (3.6) is satisfied, Stop; otherwise, go to Step 1.
```

Fig. 3.1. Flow chart of backward elimination

4 Empirical Study

In order to evaluate the performance of GPs-BE, we compare it with GPs, GPs-U, SVM, RVM and AS on four benchmark datasets, i.e. Friedman1 [10], Boston Housing, Abalone and Computer Activity [11]. GPs-U denotes GPs whose covariance function has the same scaling factors. Before experiments, all the training data are scaled in [-1, 1] and the testing data are adjusted using the same linear transformation. For Friedman1 and Boston Housing data sets, the results are averaged over 100 random splits of the full datasets. For Abalone and Computer Activity data sets, the results are averaged over 10 random splits of the mother datasets. The free parameters in GPs, GPs-BE and GPs-U are optimized by maximizing the evidence. The free parameters in RVM, SVMs and AS are selected by 10-fold cross validation procedure.

Table 4.1. Characteristics of four benchmark datasets

Abrr.	Problem	Attributes	Total Size	Training Size	Testing Size
FRI	Friedman1	10	5240	240	5000
BOH	Boston Housing	13	506	481	25
ABA	Abalone	8	4117	1000	3117
COA	Computer Activity	21	8192	1000	7192

Table 4.2. Mean of the testing errors of six algorithms

Problem	GPs	GPs-BE	GPs-U	RVM	SVMs	AS
FRI	0.46	0.47	2.62	2.84	2.68	2.80
BOH	9.23	9.31	9.67	9.92	10.66	10.02
ABA	4.61	4.65	4.63	4.62	4.72	4.68
COA	6.61	6.68	11.65	11.41	11.34	12.09

Table 4.3. Mean of the number of basis functions of six algorithms on benchmark datasets

Problem	GPs	GPs-BE	GPs-U	RVM	SVMs	AS
FRI	240.00	63.00	240.00	70.90	178.30	78.70
BOH	481.00	116.00	481.00	52.88	165.73	60.9
ABA	1000.00	15.70	1000.00	11.00	470.70	11.1
COA	1000.00	86.70	1000.00	47.80	357.90	43.6

Table 4.4. Runtime of six algorithms on benchmark datasets

Problem	GPs	GPs-BE	GPs-U	RVM	SVMs	AS
FRI	217.53	228.78	453.70	1652.05	1246.32	290.35
BOH	8012.57	8439.69	9946.76	26922.98	47005.36	7639.03
ABA	4457.84	4687.31	6922.70	13093.14	39631.06	4440.03
COA	6591.48	6849.81	7373.28	15150.28	67127.36	5183.14

From Table 4.2 we know that GPs-BE and GPs obtain similar generalization performance and are significantly better than GPs-U, RVM, SVMs and AS in the two regression tasks, i.e. Friedman1and Computer Activity. As for the remaining two tasks, all the six approaches have similar performance. Since GPs-U is often superior to SGGPs and SOGPs in terms of the generalization performance, GPs-BE is expected to have the better generalization performance than SGGPs and SOGPs.Table 4.3 show that the number of basis functions of GPs-BE approaches that of RVM and AS, and is significantly smaller than that of GPs, GPs-U and SVMs. Table 4.4 show that the runtime of GPs-BE approaches that of GPs, GPs-U and AS, and is significantly smaller than that of GPs, GPs-U and SVMs.

An alternative is to select the basis functions using the forward selection proposed by [12-13]. Table 4.5 compares our method with forward selection in the same stop criterion.

Table 4.5. Comparison of backward elimination and forward selection

Problem	Backward Elimination		Forward Selection	
FRI	0.47	63.00	0.47	69.30
BOH	9.31	116.00	9.33	131.72
ABA	4.65	15.70	4.66	20.20
COA	6.68	86.70	6.71	97.40
Normalized Mean	**0.998**	**0.864**	1.000	1.000

Table 4.5 shows that the backward elimination outperforms the forward selection in the performance and the number of basis functions in the same stop criterion.

In summary, GPs-BE greatly reduces the number of basis functions of GPs almost without sacrificing the performance and increasing the runtime. Moreover, GPs-BE is better than GPs-U in performance, which further indicates the performance of GPs-BE is better than that of SGGPs and SOGPs. GPs-BE is better than SVMs in all the three aspects. GPs-BE is also better than RVM and AS in performance with the similar number of basis functions and runtime. Finally, the backward elimination outperforms the forward selection in the same stop criterion.

5 Conclusion

This paper presents a backward elimination algorithm to select the basis functions for GPs. By integrating rank-1 update, we can implement GPs-BE at a reasonable cost. The results show that GPs-BE greatly reduces the number of basis functions of GPs almost without sacrificing the performance and increasing the runtime. Comparisons with forward selection show that GPS-BE obtains better performance and smaller basis functions in the same stop criterion.

This research is supported by National Natural Science Foundation of China under grant 60372050 and 60133010 and National “973” Project grant 2001CB1309403.

References

1. Williams, C. K. I., Rasmussen, C. E.: Gaussian Processes for Regression. Advances in Neural Information Processing Systems 8 (1996) 514-520
2. Rasmussen, C. E.: Evaluation of Gaussian Processes and Other Methods for Non-linear Regression. Ph.D. thesis, Dep.of Computer Science, University of Toronto. Available from http://www.cs.utoronto.ca/~carl/.
3. Vapnik, V.: The Nature of Statistical Learning Theory. New York: Springer-Verlag (1995)
4. Tipping, M. E.: Sparse Bayesian Learning and the Relevance Vector Machine. Journal Machine Learning Research 1 (2001) 211-244
5. Figueiredo, M. A. T.: Adaptive Sparseness for Supervised Learning. IEEE Trans. Pattern Analysis and Machine Intelligence 25 (2003) 1150-1159
6. Smola, A. J., Bartlett, P. L.: Sparse Greedy Gaussian Processes Regression, Advances in Neural Information Processing Systems 13 (2000) 619-625
7. Csato, L., Opper, M.: Sparse Online Gaussian Processes, Neural Computation, 14 (2002) 641-669
8. Berger, J. O.: Statistical Decision Theory and Bayesian Analysis. Springer, Second Edition (1985)
9. Williams, C. K. I.: Prediction with Gaussian Processes: from Linear Regression to Linear Prediction and Beyond. Learning and Inference in Graphical Models (1998) 1-17
10. Friedman, J. H.: Multivariate Adaptive Regression Splines. Annals of Statistics 19 (1991) 1-141
11. Blake, C. L., Merz, C. J.: UCI Repository of Machine Learning Databases, Technical Report, University of California, Department of Information and Computer Science, Irvine, CA (1998) Data available at http://www.ics.uci.edu/~mlearn/MLRepository.html
12. Chen, S., Cowan, C. F. N., Grant, P. M.: Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks. IEEE Trans. Neural Networks 2 (1991) 302-309
13. Bo, L. F., Wang, L., Jiao, L. C.: Sparse Bayesian Learning Based on an Efficient Subset Selection, Lecture Notes in Computer Science 3173 (2004) 264-269